

Towards Ontology-based QoS Aggregation for Composite Web Services

Paul Karaenke, Joerg Leukel

Department of Information Systems 2
University of Hohenheim
Schwerzstr. 35
70599 Stuttgart, Germany
{paul.karaenke, joerg.leukel}@uni-hohenheim.de

Abstract: Determining the QoS (quality of service) of composite Web services is of crucial importance for both service providers and consumers. However, service descriptions of constituent services are often heterogeneous which makes the aggregation of multiple single-service QoS difficult. In this paper, we address the heterogeneity problem from the perspective of knowledge engineering. We propose a QoS aggregation ontology; i.e., a formal parameter classification which can be used to determine the right aggregation for annotated QoS parameters based on workflow patterns.

1 Introduction

Composite Web services allow for realizing business processes by means of composing elementary services under a shared workflow [Du04]. To *determine the quality of service (QoS) level of a composite service*, all relevant service descriptions, i.e., QoS parameters, have to be (1) aligned to a common vocabulary and (2) aggregated depending on the composition. It is especially important for service providers who want to guarantee certain QoS levels for their customers as part of service level agreements (SLAs). This task is known as QoS aggregation [JRM04][Hwa07]. Our research addresses the problem of *heterogeneous QoS parameter descriptions*; heterogeneity restricts or even prevents computing an aggregated QoS. We study the problem from a knowledge engineering perspective. The objective is to develop a QoS aggregation ontology which can be used to determine the right aggregation for respectively annotated QoS parameters. This ontology contains a classification of QoS parameters, a set of aggregation types and their mapping to workflow patterns from workflow management research [Aa03].

The remainder is organized as follows. Section 2 develops a classification of QoS parameters and their aggregation based on existing literature. Section 3 proposes the ontology. Section 4 relates our approach to existing work. Finally, we draw conclusions and identify avenues of future research.

2 QoS Aggregation

In this section, we formalize QoS aggregation by studying its two determinants, parameter and composition. The formalism is an aggregation function for each combination of parameter and composition. Its development requires therefore three steps: (1) definition of parameters, (2) definition of composition, and (3) definition of aggregation function. The work by Jaeger et al. [JRM04] contributes core elements. We adopt these elements and extend them to arrive at more comprehensive aggregation functions. We propose a classification with regard to parameter aggregation. The rationale is that if two parameters share the same aggregation function, then they belong to the same parameter class, regardless of other characteristics such as naming or intentional semantics.

The composition can be assessed by referring to workflow patterns. Jaeger et al. [JRM04] analyzed their effect on parameter aggregation and proposed seven *composition patterns* (CP), i.e., Sequence, Loop, XOR-XOR, AND-AND, AND-DISC, OR-OR, and OR-DISC. For instance, the OR-OR pattern describes an OR-split followed by an OR-join. For each combination of parameter type and CP an aggregation function captures how to determine the QoS. Next, we limit parameters to numerical ones. Non-numerical parameters can not be mathematically aggregated by equations but require operations on characters (for string types) and sets (for enumerative type).

Since the aggregation function depends on the pair of parameter type and composition pattern, there exist $6 \cdot 4 = 24$ aggregation functions. However, a number of pairs share the same aggregation function. Therefore, we can reduce the number of functions to actually six only. We define generic aggregation functions as shown in Table 1, with x_1, \dots, x_n denoting the parameter values to be aggregated, and k for the number of iterations in a Loop pattern. In contrast to Jaeger et al., we do not consider probabilities for OR-splits; i.e., we calculate the minimal and maximal values that can occur.

Table 1: Generic aggregation functions

Aggregation function	Definition
x_{sum}	$x_{sum}(x_1, \dots, x_n) = \sum_{i=1}^n x_i$
$x_{product}$	$x_{product}(x_1, \dots, x_n) = \prod_{i=1}^n x_i$
x_{max}	$x_{max}(x_1, \dots, x_n) = \max(x_1, \dots, x_n)$
x_{min}	$x_{min}(x_1, \dots, x_n) = \min(x_1, \dots, x_n)$
x_{power}	$x_{power}(x) = x^k$
x_{linear}	$x_{linear}(x) = kx$

These generic aggregation functions are used in Table 2 to assign a function to each pair of parameter type and CP. For each pair, functions determine the maximum and minimum value of the respective parameter type; this distinction is required because of alternative control flows due to the existence of OR, XOR, or loops in patterns.

Table 2: Aggregation functions for upper and lower bounds of QoS parameters

Type	Max/Min	Se-quence	Loop	XOR-XOR	AND-AND	AND-DISC	OR-OR	OR-DISC	Example Parameter
1	max	x_{sum}	x_{linear}	x_{max}	x_{sum}	x_{sum}	x_{max}	x_{max}	cost
	min	x_{sum}	x_{linear}	x_{min}	x_{sum}	x_{sum}	x_{min}	x_{min}	
2	max	x_{sum}	x_{linear}	x_{max}	x_{max}	x_{max}	x_{max}	x_{max}	execution time
	min	x_{sum}	x_{linear}	x_{min}	x_{max}	x_{min}	x_{min}	x_{min}	
3	max	x_{min}	$x_{identity}$	x_{max}	x_{min}	x_{min}	x_{max}	x_{max}	throughput
	min	x_{min}	$x_{identity}$	x_{min}	x_{min}	x_{min}	x_{min}	x_{min}	
4	max	$x_{product}$	x_{power}	x_{max}	$x_{product}$	$x_{product}$	x_{max}	x_{max}	uptime probability
	min	$x_{product}$	x_{power}	x_{min}	$x_{product}$	$x_{product}$	$x_{product}$	$x_{product}$	

3 QoS Aggregation Ontology

In this section, we propose an ontology for QoS aggregation. The ontology incorporates the classification from section 2. It relates the identified parameter types with generic aggregation formulae and composition patterns by means of description logic (DL). Based on the ontology, reasoning determines the parameter aggregation for any given (annotated) QoS parameter and workflow pattern. We specify the ontology using the description logic *SHOIN* due to its high expressiveness. This logic is also available in the Web ontology language OWL DL. Figure 1 gives an overview of the ontology, showing its concepts and roles.

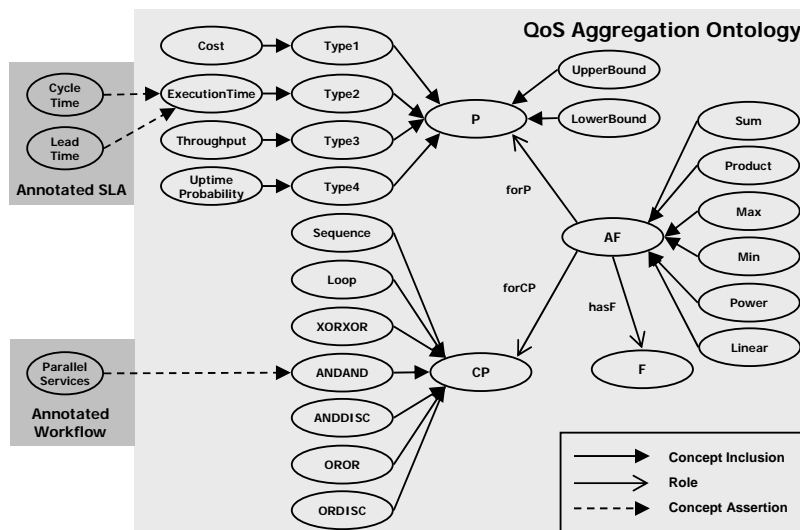


Figure 1: QoS Aggregation Ontology (Overview)

First, concept AF and its disjoint sub-concepts represents all generic aggregation functions identified in section 2. Regarding their dependence, two functional roles $forP$ and $forCP$ exist. The concept P denotes the parameter types identified in section 2 by a sub-concept each. For each type, we add at least one sub-concept for commonly used parameters such as cost. Note that these are just examples which can be easily enhanced to include also synonyms which would ease finding the right type during the annotation process. The concept CP and its disjoint sub-concepts stand for composition patterns.

Second, we need to define for which parameter type and composition pattern which function is valid. Thus, membership to the respective concept is restricted by universal restrictions over the two roles. Table 2, for instance, says that a sequence of type 1 parameters is aggregated by sum; hence concept Sum is extended by the concept definition $Sum \equiv \exists forP.Cost \sqcap \forall forCP.Sequence$. Since concept Sum is valid for more than one combination of parameter type and pattern, the concept definition consists actually of several pairs of universal restrictions being concatenated by a logical OR. Table 3 contains the full definitions of all aggregation formula concepts.

Table 3: Concept Definitions for Parameter Aggregations

Concept	Inclusion
<i>Sum</i>	$(\exists forP.Type1 \sqcap (\forall forCP.Sequence \sqcup \forall forCP.ANDAND \sqcup \forall forCP.ANDDISC)) \sqcup (\exists forP.Type2 \sqcap \forall forCP.Sequence)$
<i>Product</i>	$(\exists forP.Type4 \sqcap ((\forall forCP.Sequence \sqcup \forall forCP.ANDAND \sqcup \forall forCP.ANDDISC) \sqcup (\forall forCP.OROR \sqcap \exists forP.LowerBound) \sqcup (\forall forCP.ORDISC) \sqcap (\exists forP.LowerBound)))$
<i>Min</i>	$(\exists forP.Type1 \sqcap \exists forP.LowerBound \sqcap (\forall forCP.XORXOR \sqcup \forall forCP.OROR \sqcup \forall forCP.ORDISC)) \sqcup (\exists forP.Type2 \sqcap \forall forP.LowerBound \sqcap (\forall forCP.XORXOR \sqcup \forall forCP.ANDDISC \sqcup \forall forCP.OROR \sqcup \forall forCP.ORDISC)) \sqcup ((\exists forP.Type3 \sqcup \exists forP.Type5) \sqcap \exists forP.UpperBound \sqcap (\forall forCP.Sequence \sqcup \forall forCP.ANDAND \sqcup \forall forCP.ANDDISC)) \sqcup ((\exists forP.Type3 \sqcup \exists forP.Type5) \sqcap \exists forP.LowerBound \sqcap \forall forCP.¬Loop) \sqcup ((\exists forP.Type4 \sqcap \exists forP.LowerBound \sqcap \forall forCP.XORXOR))$
<i>Max</i>	$(\exists forP.Type1 \sqcap \exists forP.UpperBound \sqcap (\forall forCP.XORXOR \sqcup \forall forCP.OROR \sqcup \forall forCP.ORDISC)) \sqcup (\exists forP.Type2 \sqcap \forall forP.UpperBound \sqcap (\forall forCP.XORXOR \sqcup \forall forCP.ANDAND \sqcup \forall forCP.ANDDISC \sqcup \forall forCP.OROR \sqcup \forall forCP.ORDISC)) \sqcup (\exists forP.LowerBound \sqcap \forall forCP.ANDAND) \sqcup ((\exists forP.Type3 \sqcup \exists forP.Type4 \sqcup \exists forP.Type5) \sqcap \exists forP.UpperBound \sqcap (\forall forCP.XORXOR \sqcup \forall forCP.OROR \sqcup \forall forCP.ORDISC))$
<i>Power</i>	$\exists forP.Type4 \sqcap \forall forCP.Loop$
<i>Linear</i>	$(\exists forP.Type1 \sqcup \exists forP.Type2) \sqcap \forall forCP.Loop$

Finally, it would be helpful to be able to determine not only the aggregation type, but also the aggregation formula. For this purpose, the ontology provides a concept F and a functional role defined as $(AF, F):hasF$. However, the problem with $SHOIN$ and $OWL DL$ is that they do not provide constructs for expressing algebraic equations as needed. It

is not possible to represent the elements of equations in a machine-readable way (except for the naïve way of encoding equations as plain strings). This requirement is still under discussion, both from theoretical [BS98] and pragmatic perspectives [IR08].

4 Related Work

To the best of our knowledge, ontology-based QoS aggregation, which incorporates a pattern-based aggregation method, has not been studied so far. [HHS09] propose to amend each parameter in a SLA with the aggregation function and a respective aggregation process. The type is stored in a specific attribute which extends the original WS-Agreement specification. The limitation is that only hierarchical compositions – which map to the AND-AND pattern – are allowed. The same is true for [BC07]. Many researchers assume that all service providers use a predefined set of parameters only; accepting this assumption, heterogeneity is solved by standardization on the semantic level. These sets vary, however, and range from very few parameters (e.g., three in [Ca04] [MJN08], five in [Ze04] [JRM04], six in [BC07], nine in [Hwa07]) to broader sets (e.g., [OEH02]). Other work avoids the set definition problem by distinguishing types of parameters; Unger et al. argue for two types only, those representing numerical respectively enumerative values [Un08].

Workflow patterns for QoS aggregation were first proposed in [JRM04], which derives seven relevant patterns from the original patterns developed in [Aa03]. Pattern-based aggregation is also found in [Hwa07] which takes a subset of five patterns and shows that the remaining ones can be transformed into this subset. Other approaches use statecharts, which provide sufficient expressivity for most control flows in service compositions, e.g., [Ze04]. [MJN08] provides a tool for QoS aggregation from WS-BPEL workflows. Cardoso et al. represent workflows as directed graphs and compute reduction rules for sequential, parallel, conditional, fault-tolerant, loop, and network; these six rules were derived from patterns, too [Ca04].

5 Conclusions

This paper proposed ontology-based QoS aggregation. It is based on the annotation principle, as adopted in Semantic Web services, meaning that service providers maintain their local parameters while annotating them according to a shared conceptualization. We think that this approach is feasible in specific environments, such as inter-organizational business processes, also because of the relative little effort required. The ontology is quite concise, demanding only one annotation per QoS parameter.

Our proposal has some *limitations*. The parameter classification is not comprehensive; it does not guarantee that each individual non-functional parameter can be mapped to a sub-concept of P . The ontology does not consider enumerative parameter types, such as studied in [Un08]. Future work is needed, which would then extend the current ontology

by additional *P* sub-concepts. It is necessary to provide stronger evidence of the usefulness and validity of our approach.

Acknowledgment

The work presented in this paper was partly funded by the German Federal Ministry of Education and Research under the project InterLogGrid (BMBF 01IG09010E).

References

- [Aa03] van der Aalst, W.M.P.; ter Hofstede, A.H.M.; Kiepuszewski, B.; Barros, A.P.: Workflow patterns. In: Distributed and Parallel Databases, vol. 14, no. 3, 2003; pp. 5-51.
- [BC07] Blake, M.B.; Cummings, D.J.: Workflow Composition of Service Level Agreements. In: Proc. IEEE Int. Conf. Services Computing (SCC 2007), Salt Lake City, 2007; pp. 138-145.
- [BS98] Baader, F.; Sattler, U.: Description logics with concrete domains and aggregation. In: Proc. 13th European Conf. Artificial Intelligence (ECAI 98), Brighton, 1998; pp. 336-340.
- [Ca04] Cardoso, J.; Sheth, A.; Miller, J.; Arnold, J.; Kochut, K.: Quality of service for workflows and web service processes. In: Journal of Web Semantics, vol. 1, no. 3, 2004; 281-308.
- [Du04] Dustdar, S.: Web Services Workflows – Composition, Co-Ordination, and Transactions in Service-Oriented Computing. In: Concurrent Engineering, vol. 12, no. 3, 2004; pp. 237-245.
- [HHS09] Haq, I.; Huqqani, A.; Schikuta, E.: Aggregating hierarchical Service Level Agreements in Business Value Networks. In: Proc. Business Process Management Conference (BPM2009), Ulm, 2009; Springer LNCS 5701, pp. 176-192.
- [Hwa07] Hwang, S.-Y.; Wang, H.; Tang, J.; Srivastava, J.: A probabilistic approach to modeling and estimating the QoS of web-services-based workflows. In: Information Sciences: an International Journal, vol. 177, no. 23, 2007; pp. 5484-5503.
- [IR08] Iannone, L.; Rector, A.L.: Calculations in OWL. In: Proc. 5th Workshop OWL: Experiences and Directions (OWLED), Karlsruhe, 2008; CEUR-WS 432.
- [JRM04] Jaeger, M.; Rojec-Goldmann, G.; Mühl, G.: QoS Aggregation for Web Service Composition using Workflow Patterns. In: Proc. 8th IEEE Enterprise Distributed Object Computing Conf. (EDOC 2004), Monterey, 2004; pp. 149-159.
- [MJN08] Mukherjee, D.; Jalote, P.; Nanda, M.G.: Determining QoS of WS-BPEL Compositions. In: Proc. 6th Int. Conf. Service-Oriented Computing (ICSOC 2008), Sydney, 2008; Springer LNCS 5364, pp. 378-393.
- [OEH02] O'Sullivan, J.; Edmond, D.; ter Hofstede, A.H.M.: What's in a Service? In: Distributed and Parallel Databases, vol. 12, no. 2-3, 2002; pp. 117-133.
- [Un08] Unger, T.; Mauchart, S.; Leymann, F.; Scheibler, T.: Aggregation of Service Level Agreements in the Context of Business Processes. In: Proc. 12th IEEE Enterprise Distributed Object Conference (EDOC 2008), Munich, 2008; pp. S. 43-52.
- [Ze04] Zeng, L.; Benatallah, B.; Ngu, A.H.H.; Dumas, M.; Kalagnanam, J.; Chang, H.: QoS-Aware Middleware for Web Services Composition. In: IEEE Transactions on Software Engineering, vol. 30, no. 5, 2004; pp. 311-327.